Design and Analysis of Distributed, Interacting Systems (DIS)

Tutorial 01
Raphael Pham
Raphael.Pham@inf.uni-hannover.de

11.04.2013
• Each exercise earns up to 10 points.
• Exercises are usually written on paper, collected at the beginning of each lecture.
• Electronic code needs to be sent in by email to
  – Raphael.Pham@inf.uni-hannover.de
  – IMPORTANT: Subject of the email must be:
  – DIS-tut <nr.-use-two-numbers> <Prename Lastname>
  – Example: DIS-tut 01 Raphael Pham
  – NOTHING more, NOTHING else; otherwise: not registered.
• If you are looking for a good diagram program, try „Dia“
The turnstile machine

- coin slot
- cancel button
- turnstile
- coin compartment
Requirements

1. If a coin is inserted, the turnstile machine holds this coin.
2. The turnstile machine can hold up to n coins.
3. Whenever the machine holds a coin, the machine can unlock the turnstile.
4. When the turnstile is turned once, the machine puts one coin in the cash deposit box (not modeled) and locks the turnstile again.
5. Whenever the “Cancel”-Button is pressed, all coins that the turnstile machine currently holds, are returned.
Environment of the software controller

- cancel button
- controller C
- turnstile
- coin slot
- coin compartment
- coin
- money_back
Cashhandler Automaton

The diagram represents a simple automaton for cash handling. The states include:
- **empty**
- **filled**

The transitions are:
- **coin** from **empty** to **filled**
- **coin** from **filled** to **empty**
- **money_back** from **filled** to **empty**
- **cash handling** label at the bottom of the diagram.
Turnstile Automata

![Turnstile Automata Diagram](image-url)
Task 1

- Model the transition system of the controller C.
- Use a variable c to denote the counting of coins.
Task 2

• Draw the transition system TS = Turnstile || Cashhandler || Controller

• Conditions:
  – we assume, only one coin is inserted
  – we add a put-away action that completes the model; safe deposit is not modeled
Composed Automaton

![Composed Automaton Diagram]

- **empty, locked, waiting**
  - coin
  - cancel
  - money_back
- **filled, locked, waiting**
  - unlock
  - turn
- **empty, locked, cancelled**
  - coin
- **filled, unlock, action**
- **filled, locked, waiting**

**put_away**
Cheatsheet (not official!)

- Write each action set down
- Check: is pairwise and only pairwise sharing given?
- Mark each pairwise shared variable by an underline, mark exclusive variables by a curl
- Compose the starting state: write down all possible actions.
  - curly actions always trigger
  - underlined actions must be present twice in order to trigger
  - repeat